

# Distributed Systems

## 24. Clusters

Paul Krzyzanowski  
pxk@cs.rutgers.edu

# Designing highly available systems

---

- Incorporate elements of fault-tolerant design
  - Replication, TMR
- Fully fault tolerant system will offer non-stop availability
  - You can't achieve this!
- Problem: expensive!

# Designing highly scalable systems

---

## SMP architecture

### Problem:

Performance gain as  $f(\# \text{ processors})$  is sublinear

- Contention for resources (bus, memory, devices)
- Also ... the solution is expensive!

# Clustering

---

Achieve reliability and scalability by interconnecting multiple independent systems

**Cluster:** a group of standard, autonomous servers configured so they appear on the network as a single machine

*single system image*

# Ideally...

---

- Bunch of off-the shelf machines
- Interconnected on a high speed LAN
- Appear as one system to users
- Processes are load-balanced
  - May migrate
  - May run on different systems
  - All IPC mechanisms and file access available
- Fault tolerant
  - Components may fail
  - Machines may be taken down

we don't get all that (yet)

(at least not in one package)

# Clustering types

---

- Supercomputing (HPC)
  - and Batch processing
- High availability (HA)
  - and Load balancing
- High availability / High scalability

# Cluster Components

# Cluster management

---

- Software to manage **cluster membership**
  - What are the nodes in the cluster?
  - What are the *live* nodes in the cluster?
- **Quorum**
  - Number of elements that must be online for the cluster to function
  - Voting algorithm to determine whether the set of nodes has quorum (a majority of nodes to keep running)
- Keep track of quorum
  - Count cluster nodes running the cluster manager
  - If over  $\frac{1}{2}$  are active, the cluster has quorum
  - Forcing a majority avoids **split-brain**

# Interconnect

# Cluster Interconnect

---

- Provide communication between nodes in a cluster
- Often separate network from the LAN
- Sometimes known as **System Area Network (SAN)**
- Goals
  - Low latency
    - Avoid OS overhead, layers of protocols, retransmission, etc.
  - High bandwidth
    - High bandwidth, switched links
    - Avoid overhead of sharing traffic with non-cluster data
  - Low CPU overhead
  - Low cost
    - Cost usually matters if you're connecting thousands of machines

# Example High-Speed Interconnects

---

- Myricom's Myrinet
  - High-speed switch fabric for low-latency, high-bandwidth, interprocess communication between nodes
  - Lower overhead than ethernet
  - 10 Gbps bandwidth between any two nodes
  - Scales to tens of thousands of nodes
  - Example: used in IBM's Linux Cluster Solution
- Infiniband
  - Direct interconnect to CPU bridge chip
  - 2.5 – 30 Gbps
- 10 Gigabit Ethernet

# Disks

# Shared storage access

---

- If an application can run on any machine, how does it access file data?
- If an application fails over from one machine to another, how does it access its file data?
- Can applications on different machines share files?

# Network File Systems

---

- One option
  - network file systems: NFS, SMB, AFS, AFP, etc.
  - Caching may cause inconsistencies
  - Data sharing is usually a problem
  - Performance may be a problem (LAN vs. local bus access)

# Shared disk

---

- Shared disk
  - Allows multiple systems to share access to disk drives
  - Works well if there isn't much contention
  - Disk access must be synchronized
    - Synchronization via a **distributed lock manager (DLM)**
- **Cluster File System**
  - Client runs a file system accessing a shared disk at the **block level**
  - Examples:  
IBM General Parallel File System (GPFS), Microsoft Cluster Shared Volumes (CSV), Oracle Cluster File System (OCFS), Red Hat Global File System (GFS)

# Cluster File System

---

- No client/server roles, no disconnected modes
- All nodes are peers and access a shared disk(s)
- **Distributed Lock Manager**
  - Process to ensure mutual exclusion when needed
  - Not needed for local file systems on shared disk
  - Inode-based locking and caching control
- **Linux GFS (no relation to Google GFS)**
  - Cluster file system accessing storage at a block level
  - Cluster Logical Volume Manager (CLVM): volume management of cluster storage
  - Global Network Block Device (GNBD): block level storage access over ethernet: cheap way to access block-level storage

# Shared nothing

---

- Shared nothing
  - No shared devices
  - Each system has its own storage resources
  - No need to deal with DLMS
  - If a machine A needs resources on B, A sends a message to B
    - If B fails, storage requests have to be switched over to a live node
  - Exclusive access to shared storage
    - Multiple nodes may have access to shared storage
    - Only one node is granted exclusive access
    - Exclusive access changed on failover

# SAN: Node-Disk interconnect

---

- Storage Area Network (SAN)
- Separate network between nodes and storage arrays
  - Fibre channel
  - iSCSI
- Any node can be configured to access any storage through a fibre channel switch
  
- **DAS**: Direct Attached Storage
- **SAN**: block-level access to a disk via a network
- **NAS**: file-level access to a remote file system

# Failover

# HA issues

---

- How do you detect failover?
- How long does it take to detect?
- How does a dead application move/restart?
- Where does it move to?

# Heartbeat network

---

- Machines need to detect faulty systems
  - Heartbeat: “ping” mechanism
- Need to distinguish system faults from network faults
  - Useful to maintain redundant networks
  - Send a periodic heartbeat to test a machine’s liveness
  - Watch out for split-brain!
- Ideally, use a network with a bounded response time
  - Microsoft Cluster Server supports a dedicated “private network”
    - Two network cards connected with a pass-through cable or hub
  - SAN interconnect
  - Cluster interconnect

# Failover Configuration Models

---

- **Active/Passive ( $N+M$  nodes)**
  - $M$  dedicated failover node(s) for  $N$  active nodes
  - Passive nodes do nothing until they're needed
  
- **Active/Active**
  - Failed workload goes to remaining nodes

# Design options for failover

---

- **Cold failover**
  - Application restart
- **Warm failover**
  - Restart last checkpointed image
  - Relies on application checkpointing itself periodically
  - May use writeahead log (tricky)
- **Hot failover**
  - Application state is lockstep synchronized
  - Ready to run immediately
  - May be difficult, expensive (resources), prone to software faults

# Design options for failover

---

- With either type of failover ...
- **Multi-directional failover**
  - Failed applications migrate to / restart on available systems
- **Cascading failover**
  - If the backup system fails, application can be restarted on another surviving system

# IP Address Takeover (IPAT)

---

- Depending on the deployment:
  - IP addresses of services don't matter. A load balancer, name server, or coordinator will identify the correct machine
  - An node in an active/passive configuration may need to take over the IP address of a failed node
  - MAC address takeover may be needed if we cannot guarantee that other nodes will flush their ARP cache
  - A node an active/active configuration may need to listen on multiple IP addresses

# Hardware support for High Availability

---

- Hot-pluggable devices
  - Minimize downtime for component swapping
- Redundant devices
  - Redundant power supplies
  - Parity on memory
  - Mirroring on disks (or RAID for HA)
  - Switchover of failed components
- Diagnostics
  - On-line serviceability

# Fencing

---

- Method of isolating a node from a cluster
  - Failed node
  - Disconnect I/O to ensure data integrity
- Types
  - Power fencing: shut power off a node
  - SAN fencing: disable a Fibre Channel port to a node
  - Disable access to a global network block device (GNBD) server

# Cluster software hierarchy

---

## Example: Windows Server clustering

- Top tier: cluster abstractions
  - Failover manager, resource monitor, cluster registry
- Middle tier: distributed operations
  - Global status update, membership, quorum (keeps track of minimal membership for functioning and who's in charge)
- Bottom tier: OS and drivers
  - Cluster disk driver, cluster network drivers
  - IP address takeover

# High Performance Computing (HPC)

# Supercomputing clusters

---

- Target complex, typically scientific, applications:
  - Large amounts of data
  - Lots of computation
  - Parallelizable application
- Many custom efforts
  - Typically Linux + message passing software + remote exec + remote monitoring

# Programming tools: MPI

---

- Message Passing Interface
- API for sending/receiving messages
  - Optimizations for shared memory & NUMA
  - Group communication support
- Other features:
  - Scalable file I/O
  - Dynamic process management
  - Synchronization (barriers)
  - Combining results

# Programming tools: PVM

---

- Software that emulates a general-purpose heterogeneous computing framework on interconnected computers
- Present a view of virtual processing elements
  - Create tasks
  - Use global task IDs
  - Manage groups of tasks
  - Basic message passing

# Clustering for performance

---

- Example: One popular effort
  - Beowulf
    - Initially built to address problems associated with large data sets in Earth and Space Science applications
    - From Center of Excellence in Space Data & Information Sciences (CESDIS), division of University Space Research Association at the Goddard Space Flight Center

# What makes it possible

---

- Commodity off-the-shelf computers are cost effective
- Publicly available software:
  - Linux, GNU compilers & tools
  - MPI (message passing interface)
  - PVM (parallel virtual machine)
- Low cost, high speed networking
- Experience with parallel software
  - Difficult: solutions tend to be custom

# What can you run?

---

- Programs that do not require fine-grain communication
- Nodes are dedicated to the cluster
  - Performance of nodes not subject to external factors
- Interconnect network isolated from external network
  - Network load is determined only by application
- Global process ID provided
  - Global signaling mechanism

# Beowulf configuration

---

- Includes:
  - BPROC: Beowulf distributed process space
    - Start processes on other machines
    - Global process ID, global signaling
  - Network device drivers
    - Channel bonding, scalable I/O
  - File system (file sharing is generally not critical)
    - NFS root
    - unsynchronized
    - synchronized periodically via rsync

# Beowulf programming tools

---

- PVM and MPI libraries
- Distributed shared memory
  - Page based: software-enforced ownership and consistency policy
- Cluster monitor
- Global *ps*, *top*, *uptime* tools
  
- Process management
  - Batch system
  - Write software to control synchronization and load balancing with MPI and/or PVM
  - Preemptive distributed scheduling: not part of Beowulf (two packages: **Condor** and **Mosix**)

# Another example

---

- Rocks Cluster Distribution
  - Employed on over 1,300 clusters
  - Mass installation is a core part of the system
    - Mass re-installation for application-specific configurations
  - Front-end central server + compute & storage nodes
  - Based on CentOS Linux
  - Rolls: collection of packages
    - Base roll includes: PBS (portable batch system), PVM (parallel virtual machine), MPI (message passing interface), job launchers, ...

# Another example

---

- Microsoft HPC Server 2008
  - Windows Server 2008 + clustering package
  - Systems Management
    - Management Console: plug-in to System Center UI with support for Windows PowerShell
    - RIS (Remote Installation Service)
  - Networking
    - MS-MPI (Message Passing Interface)
    - ICS (Internet Connection Sharing) : NAT for cluster nodes
    - Network Direct RDMA (Remote DMA)
  - Job scheduler
  - Storage: iSCSI SAN and SMB support
  - Failover support

# Batch Processing

# Batch processing

---

- Common application: graphics rendering
  - Maintain a queue of frames to be rendered
  - Have a dispatcher to remotely exec process
- Virtually no IPC needed
- Coordinator dispatches jobs

# Single-queue work distribution

---

- Render Farms:
  - Pixar:
    - 12,500 cores on Dell render blades running Linux and Renderman
    - Custom Linux software for articulating, animating/lighting (Marionette), scheduling (Ringmaster), and rendering (RenderMan)
    - Cars 2: average frame took 11.5 hours to Render (vs. 8 hours for Cars).
  - DreamWorks:
    - >3,000 servers and >1,000 Linux desktops  
HP xw9300 workstations and HP DL145 G2 servers with 8 GB/server
    - Kung Fu Panda 2 used 100 TB data and required over 55 million render hours
    - Shrek 3: 20 million CPU render hours. Platform LSF used for scheduling + Maya for modeling + Avid for editing+ Python for pipelining – movie uses 24 TB storage

[http://news.cnet.com/8301-13772\\_3-20068109-52/new-technology-revs-up-pixars-cars-2/](http://news.cnet.com/8301-13772_3-20068109-52/new-technology-revs-up-pixars-cars-2/)

# Single-queue work distribution

---

- Render Farms:
  - ILM:
    - 3,000 processor (AMD) renderfarm; expands to 5,000 by harnessing desktop machines
    - 20 Linux-based SpinServer NAS storage systems and 3,000 disks from Network Appliance
    - 10 Gbps ethernet
  - Sony Pictures' Imageworks:
    - Over 1,200 processors
    - Dell and IBM workstations
    - almost 70 TB data for Polar Express

# Batch Processing

---

- OpenPBS.org:
  - Portable Batch System
  - Developed by Veridian MRJ for NASA
- Commands
  - Submit job scripts
    - Submit interactive jobs
    - Force a job to run
  - List jobs
  - Delete jobs
  - Hold jobs

# Load Balancing

# Functions of a load balancer

---

- Load balancing
- Failover
- Planned outage management

# Redirection

---

Simplest technique

HTTP REDIRECT error code

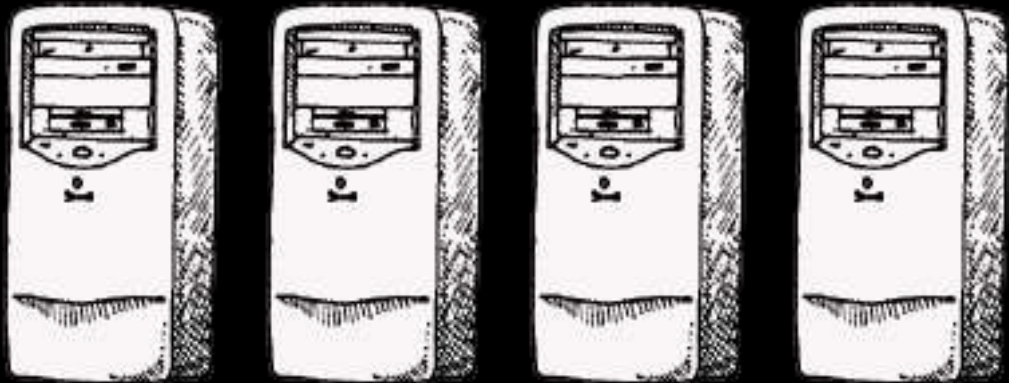
# Redirection

---

Simplest technique

HTTP REDIRECT error code

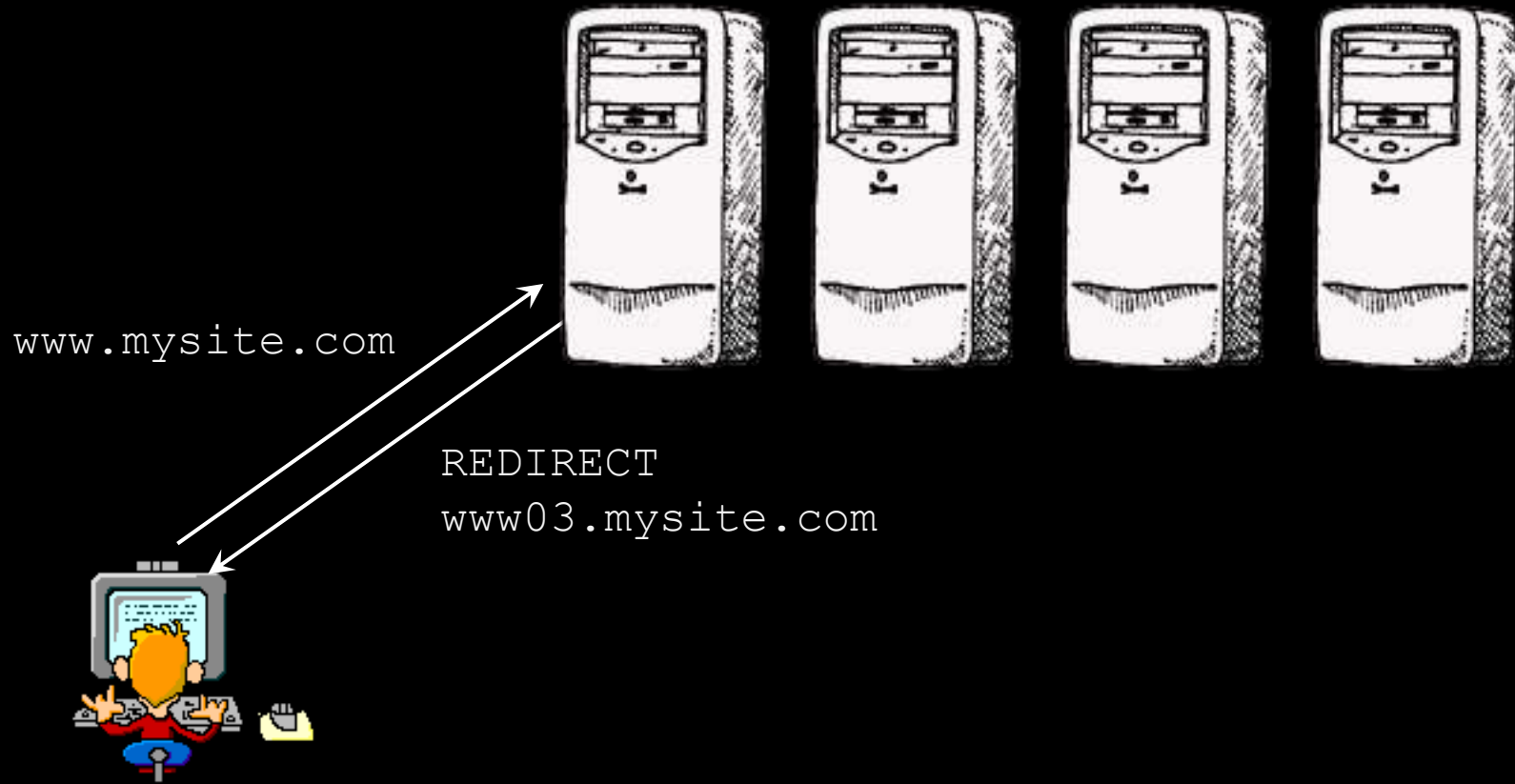
`www.mysite.com`



# Redirection

Simplest technique

HTTP REDIRECT error code

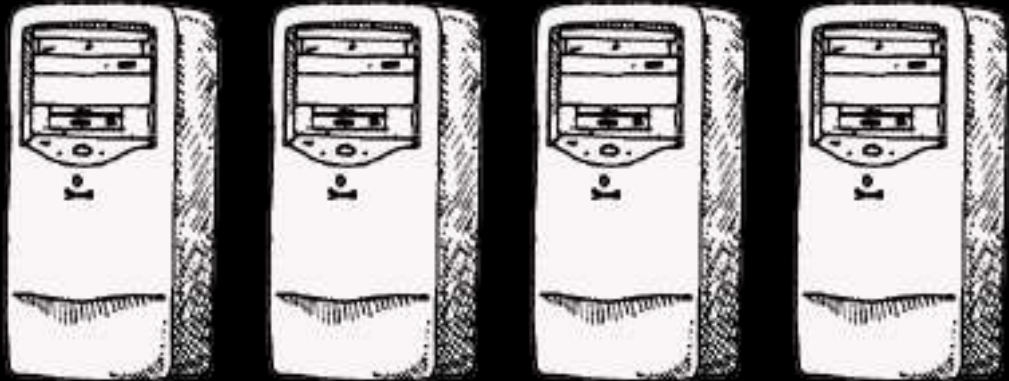


# Redirection

---

Simplest technique

HTTP REDIRECT error code



[www03.mysite.com](http://www03.mysite.com)

# Redirection

---

- Trivial to implement
- Successive requests automatically go to the same web server
  - Important for sessions
- Visible to customer
  - Some don't like it
- Bookmarks will usually tag a specific site

# Software load balancer

---

e.g.: IBM Interactive Network Dispatcher Software

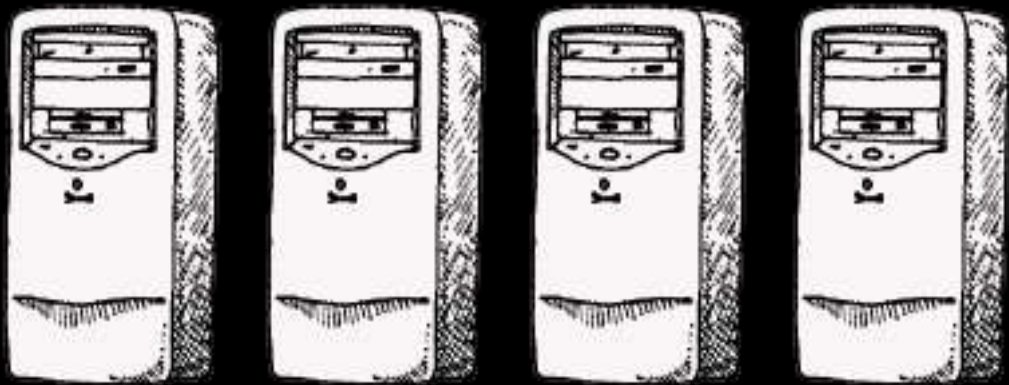
## Forwards request via load balancing

- Leaves original source address
- Load balancer not in path of outgoing traffic (high bandwidth)
- Kernel extensions for routing TCP and UDP requests
  - Each client accepts connections on its own address and dispatcher's address
  - Dispatcher changes MAC address of packets.

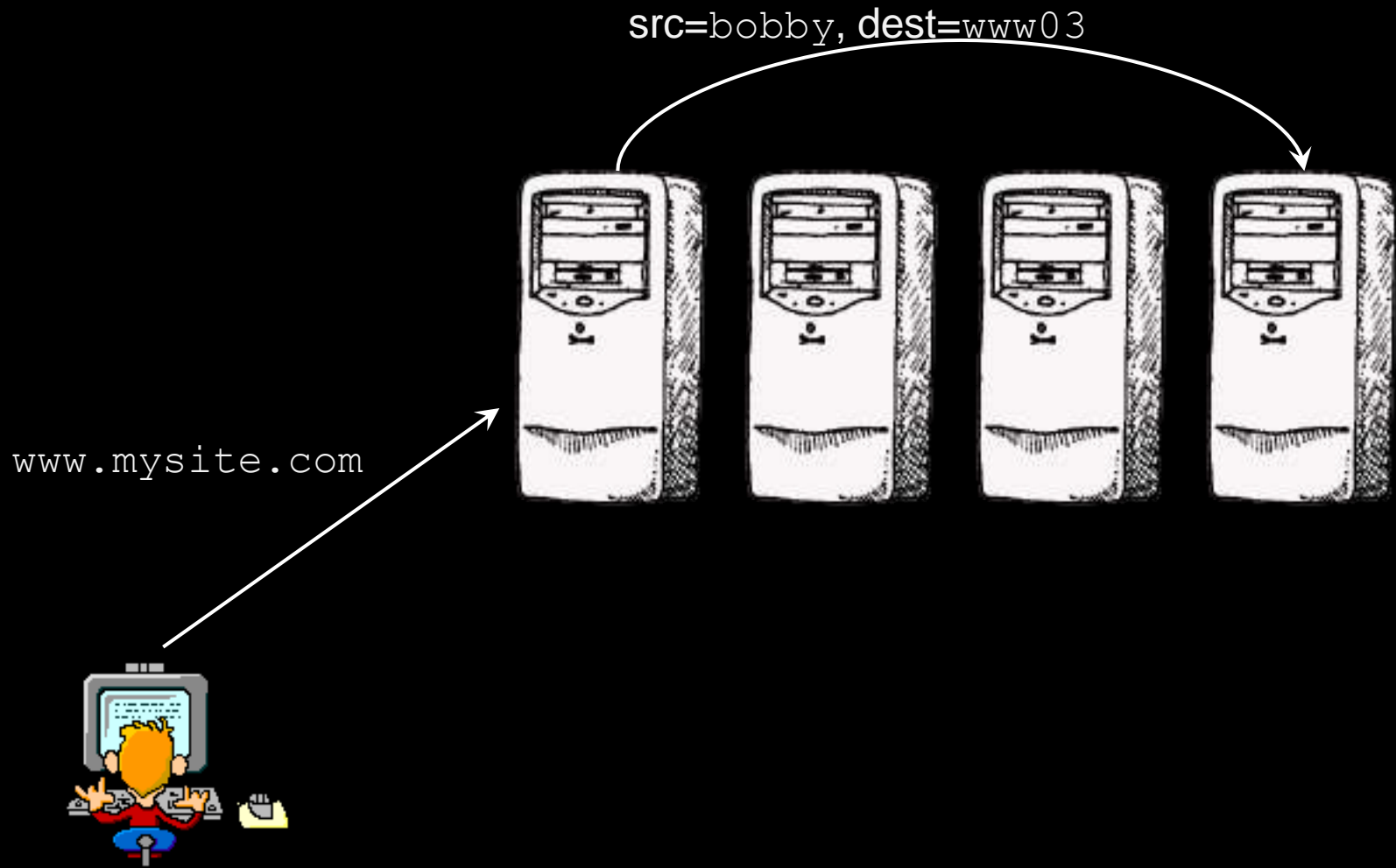
# Software load balancer

---

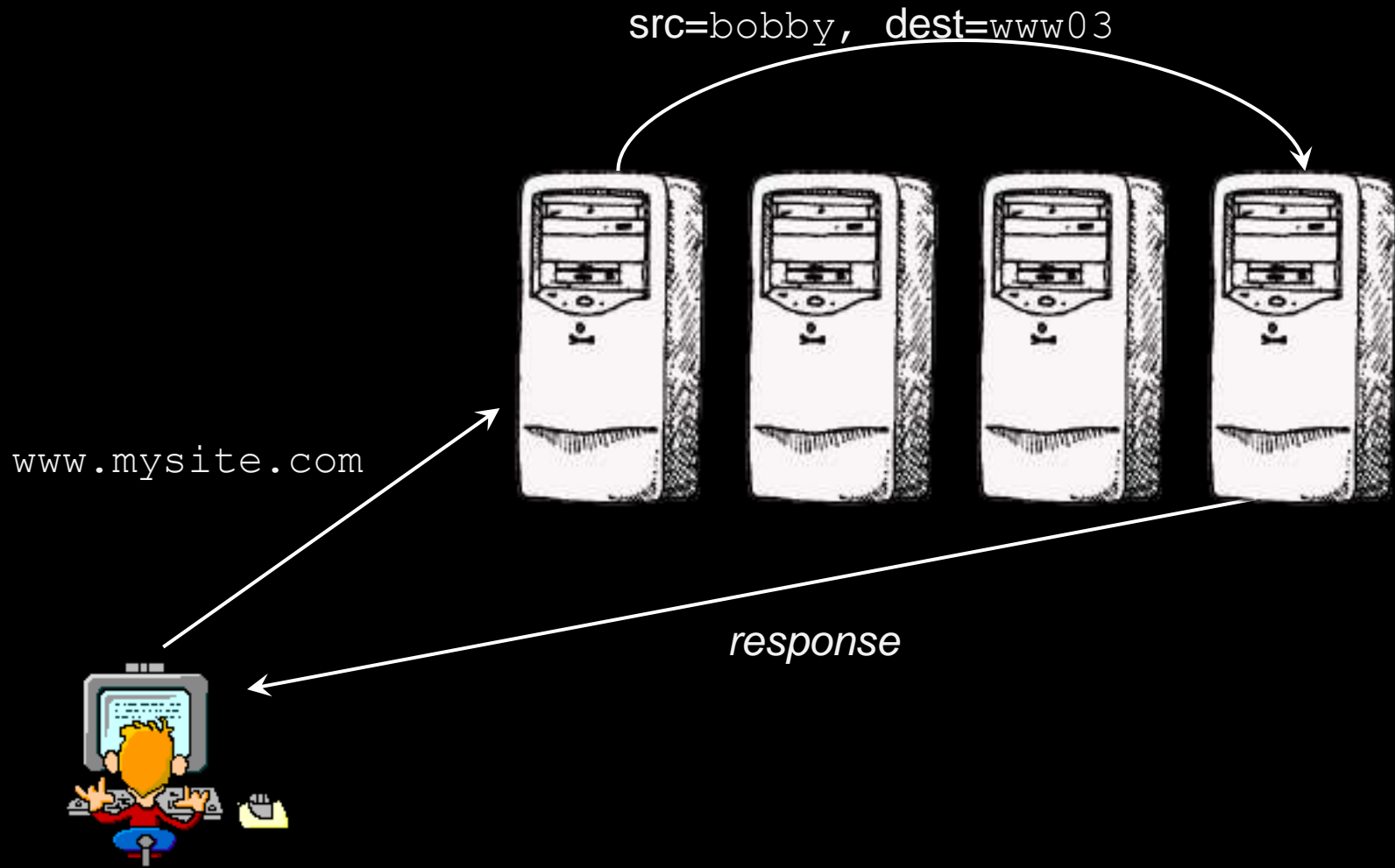
www.mysite.com



# Software load balancer



# Software load balancer



# Load balancing router

---

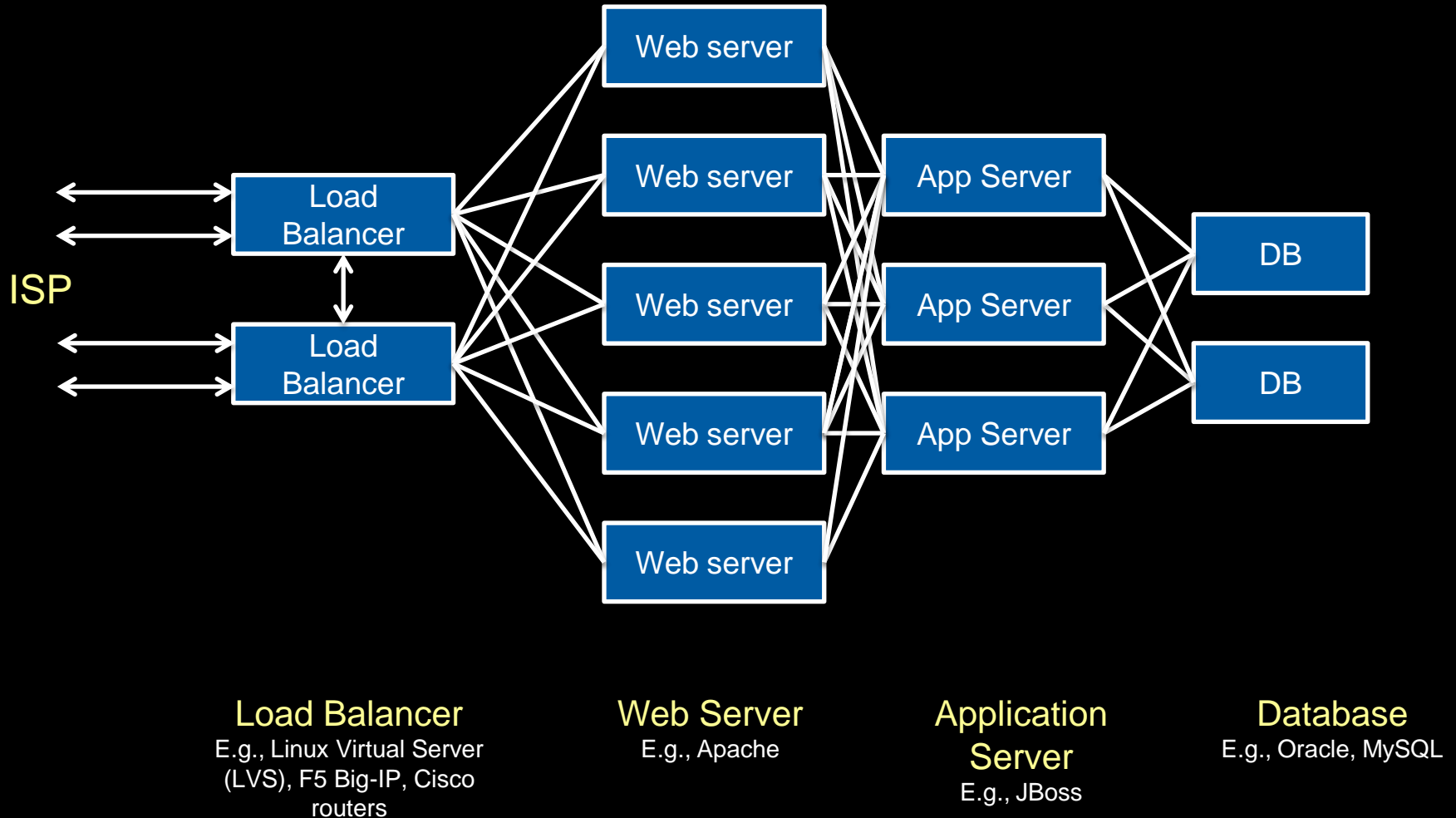
- Routers have been getting smarter
  - Not just simple packet forwarding
  - Most support packet filtering
  - Add load balancing
- Most Cisco routers, Altheon, F5 Big-IP

# Load balancing router

---

- Assign one or more virtual addresses to physical address
  - Incoming request gets mapped to physical address
- Special assignments can be made per port
  - e.g. all FTP traffic goes to one machine
- Balancing decisions:
  - Pick machine with least # TCP connections
  - Factor in weights when selecting machines
  - Pick machines round-robin
  - Pick fastest connecting machine (SYN/ACK time)

# Load Balancing



The End